◢ Aspire Engineering Coding Project

Problem Statement

We all rely on open-source libraries and repositories, but keeping track of updates, versions, and relevant changes can be challenging. Your task is to create a **web application** that helps users track GitHub repositories and view their latest activity.

This app will include a **backend** powered by GraphQL and PostgreSQL, with a frontend built using **TypeScript** and **React**.

Project Requirements

Core Requirements

1. **Frontend**
   ○ Create a React app using TypeScript.
   ○ Provide an intuitive and responsive UI for managing a list of GitHub repositories.
   ○ Allow users to:
      ■ Add repositories they want to track.
      ■ View the latest release information (e.g., version, release date).
      ■ Mark releases as "seen" to indicate updates have been reviewed.
      ■ Identify repositories with unseen updates via visual indicators.
   ○ Integrate a GraphQL API to fetch and manipulate data.
   ○ Handle client-side caching for better performance.
2. **Backend**
   ○ Use Node.js with a GraphQL framework (e.g., Apollo Server or GraphQL Yoga).
   ○ Implement a PostgreSQL database to store:
      ■ Tracked repositories.
      ■ Release metadata.
      ■ User-specific seen status for each release.
   ○ Fetch data from the GitHub API using Octokit (or similar libraries) and expose it via the GraphQL API.
3. **Data Synchronization**
   ○ Implement periodic updates of repository data to capture new releases.
   ○ Provide a "Refresh" button to manually reload data from the GitHub API.
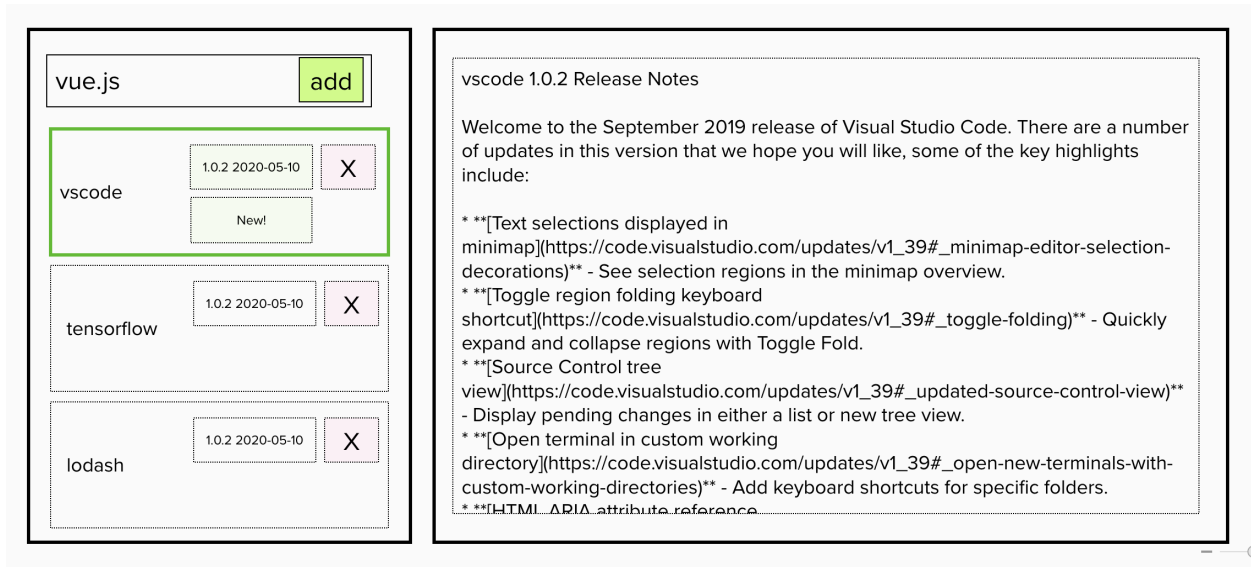

Tech Stack

● **Frontend**: React, TypeScript, Apollo Client (or an alternative GraphQL client).
● **Backend**: Node.js, GraphQL (Apollo Server or similar), Octokit (GitHub API).

Aspire Engineering Coding Project

- **Database**: PostgreSQL.

Example:


```
vue.js                      add

vscode      1.0.2 2020-05-10    X

            New!

tensorflow  1.0.2 2020-05-10    X

lodash      1.0.2 2020-05-10    X
```

```
vscode 1.0.2 Release Notes

Welcome to the September 2019 release of Visual Studio Code. There are a number
of updates in this version that we hope you will like, some of the key highlights
include:

* **[Text selections displayed in
minimap](https://code.visualstudio.com/updates/v1_39#_minimap-editor-selection-
decorations)** - See selection regions in the minimap overview.
* **[Toggle region folding keyboard
shortcut](https://code.visualstudio.com/updates/v1_39#_toggle-folding)** - Quickly
expand and collapse regions with Toggle Fold.
* **[Source Control tree
view](https://code.visualstudio.com/updates/v1_39#_updated-source-control-view)**
- Display pending changes in either a list or new tree view.
* **[Open terminal in custom working
directory](https://code.visualstudio.com/updates/v1_39#_open-new-terminals-with-
custom-working-directories)** - Add keyboard shortcuts for specific folders.
* **[HTML ARIA attribute reference
```

MVP User Stories

1. **Track Repositories**:
   - Users can add GitHub repository URLs to track their updates.
   - Data persists across sessions (e.g., stored in PostgreSQL).
2. **Latest Release Details**:
   - Display repository name, description, and the latest release version and date.
   - Use a visual indicator for new updates since the last "seen" action.
3. **Mark as Seen**:
   - Users can mark a release as "seen."
   - Repositories with unseen updates should be visually distinct.
4. **Data Reload**:
   - Users can manually refresh the repository list to fetch the latest data.

Stretch Goals (Optional)

- **Release Details**: Include release notes and commit history in a detailed view.
- **Filters and Sorting**: Allow users to filter or sort repositories (e.g., by update status, date, or repository name).
- **Mobile Responsiveness**: Ensure the app is fully functional on mobile devices.
- **Notifications**: Implement desktop or in-app notifications for new updates.

◢ Aspire Engineering Coding Project

- **Authentication**: Add basic authentication (e.g., GitHub OAuth) to personalize tracked repositories for specific users.
- **Data Sync Optimization**: Use webhooks (via GitHub API) or background jobs for real-time update tracking.
- **Local Development Server**: Provide a mock GraphQL server with test data for rapid iteration without needing the GitHub API.

Submission Guidelines

1. Submit your project as a public GitHub repository. Include:
   - A README with setup instructions.
   - Steps to run the app locally.
   - Notes on what's implemented and any trade-offs made.
   - Suggestions for future improvements if given more time.
2. Spend a reasonable amount of time (e.g., 2-3 hours). It's okay to submit a partial implementation as long as it showcases your approach.
3. Ensure the project is Demo-able (from your local is ok).
4. Feel free to be creative with the UI and project structure. The goal is to demonstrate your technical and design skills.

Resources

- **GitHub API Documentation**: GitHub REST API and GraphQL API.
- **Octokit Libraries**: Octokit.
- **GraphQL Resources**: Apollo Server, GraphQL Yoga.
- **PostgreSQL**: PostgreSQL Documentation.
- **Frontend Libraries**: React, Material-UI, or other modern UI libraries.